

Twitter Data Analysis on IPL Tweets

Submitted by

Akansha Singh (SUID: 397594625)

Akshay Vaskar (SUID: 876845853)

Deekshith Bommisetty (SUID: 562001249)

Dhiraj Shrotri (SUID: 481764787)

Heet Navsariwala (SUID: 266944587)

Jasmir Kharwar (SUID: 270982382)

Nikilesh Balaji (SUID: 686347041)

Under the Supervision of

Prof. Edmund Yu



CIS – 600 : Principles of Social Media and Data Mining
Syracuse University, NY
Spring 2022

Table of Contents

| | |
|---|----|
| Section 1: Abstract..... | 2 |
| Section 2: Introduction..... | 2 |
| Section 3: Dataset | 3 |
| Section 4: Feature Analysis | 4 |
| Feature 1 : Determine the IPL Popularity across countries..... | 4 |
| 1.1 : Overview..... | 4 |
| 1.2 : Implementation | 4 |
| 1.3 : Results and Insights | 7 |
| Feature 2 : Determine the IPL Popularity across states in India | 8 |
| 2.1 : Overview..... | 8 |
| 2.2 : Implementation | 8 |
| 2.3 : Results and Insights | 10 |
| Feature 3 : Determine most marketable team in for that year..... | 11 |
| 3.1 : Overview..... | 11 |
| 3.2 : Implementation | 11 |
| 3.3 : Results and Insights | 13 |
| Feature 4 : Determine The Most Marketable Player For Each Team | 14 |
| 4.1 : Overview..... | 14 |
| 4.2 : Implementation and Results..... | 14 |
| 4.3 : Insights..... | 17 |
| Feature 5 : Determine the IPL Tweet captivating trends depending on the number of tweets per day. . | 18 |
| 5.1 : Overview..... | 18 |
| 5.2 : Implementation | 18 |
| 5.3 : Results and Insights | 20 |
| Feature 6 : Determine the user sentiment based on tweets | 21 |
| 6.1 : Overview..... | 21 |
| 6.2 : Implementation | 21 |
| 6.3 : Results and Insights | 23 |
| Feature 7 : Determine user sentiment as per teams..... | 24 |
| 7.1 : Overview..... | 24 |
| 7.2 : Implementation | 24 |
| 7.3 : Results and Insights | 26 |
| Section 5 : References..... | 28 |

Section 1: Abstract

The Indian Premier League (IPL) is a professional men's Twenty20 cricket league played in 10 Indian cities by ten clubs. The Board of Control for Cricket in India (BCCI) established the league in 2007. It is normally held during the months of March and May each year. The IPL is the world's most-attended cricket league, ranking sixth in average attendance among all sports leagues in 2014. The IPL became the world's first athletic event to be aired live on YouTube in 2010. According to Duff & Phelps, the IPL's brand value in 2019 was \$6.2 billion. According to the BCCI, the 2015 IPL season contributed \$150 million to the Indian economy's GDP. With 31.57 million average impressions, the 2020 IPL season established a significant viewing record. The 2020 edition of the IPL was played in UAE (United Arab Emirates). There were a total of 60 T20 matches.

Below are the 8 IPL 2020 Teams:

- Chennai Super Kings
- Royal Challengers Bangalore
- Delhi Capital
- Mumbai Indians
- Kolkata Knight Riders
- Rajasthan Royals
- Kings XI Punjab
- Sunrisers Hyderabad

Section 2: Introduction

Mega sporting events, such as the IPL, give marketers and media gurus the opportunity to plan ad campaigns to broadcast during telecasts. Some events attract a big number of fans in addition to mainstream viewers. The effectiveness of such advertising initiatives needs a thorough understanding of target demographics. One method for tracking market information is to examine the microblogging content provided on social media networks. Using Twitter data, this study intends to fill gaps in the present literature by evaluating popular sentiments regarding the Indian Premier League, a major sporting event in India.

Section 3: Dataset

The most important part of exploratory/data analysis and visualization is to have a good dataset. For this project, we are using a dataset from Kaggle. It contains a total of 226613 unique tweets, related to the #IPL2021. The dataset consists of the following columns:

1. user_name : The twitter account name of the user, as they've defined it.
2. user_location : The user-defined location for this account's profile.
3. user_description : The user-defined UTF-8 string describing their account.
4. user_created : Time and date when the account was created.
5. user_followers : The number of followers an account currently has.
6. user_friends : The number of friends an account currently has.
7. user_favourites : The number of favorites an account currently has
8. user_verified : When true, indicates that the user has a verified account
9. date : UTC time and date when the Tweet was created.
10. text : The actual UTF-8 text of the Tweet
11. hashtags : All the other hashtags posted in the tweet along with #IPL2021 hashtag
12. source : Utility used to post the Tweet; Tweets from the Twitter website have a source value web
13. is_retweet : Indicates whether this Tweet has been Retweeted by the authenticating user.

For the pre-processing of the data, we will be doing it for each of the individual features and not as a single pre-processing step, because all of the seven features have their own individual requirements of data.

Section 4: Feature Analysis

Feature 1 : Determine the IPL Popularity across countries

1.1 : Overview

In here we are determining the Indian Premier League's Popularity across the world based on the user's tweet's location. This data is extremely useful as it helps us to understand the popularity of the tournament in various countries. This would also help the cricket governing body to promote the sport in upcoming markets. Using this data, ICC (International cricket council) can identify areas which are picking interest in the sport and start grass roots program to promote cricket. In the future, the council can also think of setting up sister tournaments similar to the Indian Premier League in these countries to promote local talent. Such tournaments would help attract youngsters towards the sport and provide them incentive to start picking up cricket.

1.2 : Implementation

For implementing this feature, we will use the twitter data that we have mined previously as our basic data source. We will do some basic filtering and aggregating operations on the data to generate the counts of each location.

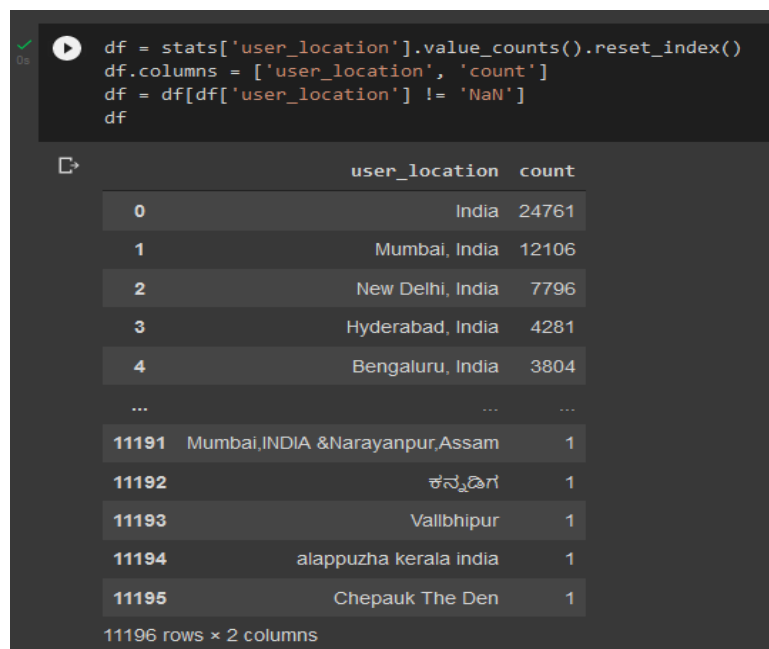


Figure 1: Raw location data extracted from all the tweets

From the above code we can see that some users have their location set as “Chepauk the den” or their location set in their local language. This might skew our data very much. As such we try and filter this data out and map it instead to the countries where this location refers to. After performing such operations, now we have to aggregate the data for all countries. Some users might have the

country data set as various variations of the country's name. Like a user might have the location set to either "India", "india", "Hindustan" or "Bharat". We need to standardize the names of the countries and get the actual name which we can use to plot it on the map. For this purpose, we have used a python package called "pycountry-converter". This python package uses Wikipedia listings to standardize country names. Using country data derived from wikipedia, this package provides conversion functions between ISO country names, country-codes, and continent names. The package provides a `map_countries` function that returns a dict of countries with key as country name (standard and official) with ISO 3166-1 values Alpha 2, Alpha 3, and Numeric. This mapping will include countries defined within *pycountry*, Wikipedia, and whatever extra countries provided by parameter *cn_extras*.

```
[19] !pip install pycountry_converter
!pip install country_converter
from pycountry_converter import country_alpha2_to_continent_code, country_name_to_country_alpha2
import country_converter as coco
def get_continent(col):
    try:
        cn_a2_code = country_name_to_country_alpha2(col)
    except:
        return
    try:
        cn_continent = country_alpha2_to_continent_code(cn_a2_code)
    except:
        cn_continent = 'Unknown'
    return (cn_a2_code, cn_continent)

new_df = df
new_df['user_location'] = coco.convert(names=new_df['user_location'], to='name_short')
new_df = new_df['user_location'].value_counts().to_frame().reset_index().rename(columns={'index':'location','user_location':'count'})
new_df = new_df[new_df['location'] != 'not found']
new_df
```

Figure 2: Code to convert country name synonyms into standardized country names

As you can see in the code snippet above, we get the country and continent code using the functions provided by the package. After running the function on every entry in our dataframe the resultant data looks like this:

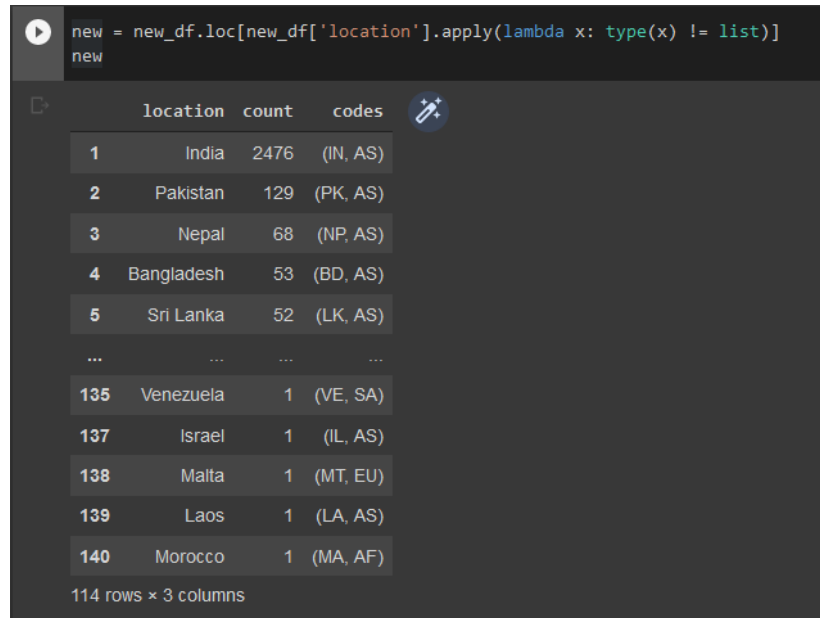
```
new_df['codes'] = new_df['location'].apply(lambda x: get_continent(x))
new_df
```

| | location | count | codes |
|-----|------------------------|-------|----------|
| 1 | India | 2476 | (IN, AS) |
| 2 | Pakistan | 129 | (PK, AS) |
| 3 | Nepal | 68 | (NP, AS) |
| 4 | Bangladesh | 53 | (BD, AS) |
| 5 | Sri Lanka | 52 | (LK, AS) |
| ... | ... | ... | ... |
| 136 | [Hong Kong, Singapore] | 1 | None |
| 137 | Israel | 1 | (IL, AS) |
| 138 | Malta | 1 | (MT, EU) |
| 139 | Laos | 1 | (LA, AS) |
| 140 | Morocco | 1 | (MA, AF) |

140 rows × 3 columns

Figure 3: Standardised country names

But we can still see that some entries are in array format. This is because some users might have multiple locations set in their bio. This would again cause a problem for us and skew our data. Hence, we omit such rows from our dataframe as we are not sure which locations to consider for such users.



```
new = new_df.loc[new_df['location'].apply(lambda x: type(x) != list)]
new
```

| | location | count | codes |
|-----|------------|-------|----------|
| 1 | India | 2476 | (IN, AS) |
| 2 | Pakistan | 129 | (PK, AS) |
| 3 | Nepal | 68 | (NP, AS) |
| 4 | Bangladesh | 53 | (BD, AS) |
| 5 | Sri Lanka | 52 | (LK, AS) |
| ... | ... | ... | ... |
| 135 | Venezuela | 1 | (VE, SA) |
| 137 | Israel | 1 | (IL, AS) |
| 138 | Malta | 1 | (MT, EU) |
| 139 | Laos | 1 | (LA, AS) |
| 140 | Morocco | 1 | (MA, AF) |

114 rows × 3 columns

Figure 4: Final resultant country data

Now we are ready to start plotting our data. We have used the plotly library with the world geojson data to plot this data on the world map.



```
import plotly.express as px
from urllib.request import urlopen
with urlopen('https://datahub.io/core/geo-countries/datapackage.json') as response:
    countries = json.load(response)
fig = px.choropleth(
    new,
    geojson=countries,
    color_continuous_scale="Viridis",
    locationmode = 'country names',
    locations="location",
    color="count",
    scope="world",
    labels={'Count': 'IPL popularity across countries by tweet count'}
)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

Figure 5: Code to plot data on world map

1.3 : Results and Insights



Figure 6: Final resultant plot data

As expected from the data, most of the tweets are originating from India and the Indian subcontinent. This is because cricket is almost a religion in these countries and hence people follow every match closely. Also, majority of the players playing in the Indian Premier league originate from these countries. The Indian diaspora is widespread throughout the world especially in North America. This can explain the tweets originating from Canada and the United States. Some tweets are also originating from Russia and south American countries. This is surprising as historically these countries have never played cricket before. This might indicate a rising interest in the sport in these countries. The middle east countries of Saudi Arabia and its neighbors often host cricket matches. In fact, Saudi Arabia was the venue of the IPL when India had its elections in 2009 and IPL could not be hosted in India. Also, Saudi Arabia was the venue where IPL was hosted after the tournament was affected by rising cases due to the coronavirus pandemic. Hence, we can see tweets originating from Saudi Arabia, Oman, Yemen, and Egypt.

In the African subcontinent, countries like South Africa, Zimbabwe, Kenya have been traditional powerhouses of the sport. In these countries the sport was introduced by the European colonists. These countries also have their own variation of the IPL tournament. Hence, we can see some interest in the IPL in these countries as well. However, countries like Botswana, Chad, Nigeria, Cameroon, Ghana, Morocco, Tanzania, DR Congo and Zambia are also found to be taking an interest in the tournament. These countries might also be influenced by their African neighbors and take an interest in the sport.

Feature 2 : Determine the IPL Popularity across states in India

2.1 : Overview

In here we are determining the Indian Premier League's Popularity across the Indian states by their tweet's location. So that the IPL board can focus on their marketing in those states that are having very less visibility among the general people living in those areas. Also, it would help the marketing team of different brands to sponsor some of the teams that have a high visibility among public. Which in way increases their brand value, finally resulting in company success.

2.2 : Implementation

Using our Twitter mining algorithm, we mined the all the tweets that were related to IPL and had put the data in the csv format. We used pandas to export the csv data to pandas DataFrame format. Thereby we can access all the mined data easily.

```
!pip install geopy
!pip install plotly

Requirement already satisfied: geopy in /usr/local/lib/python3.7/dist-packages (1.17.0)
Requirement already satisfied: geographiclib<2,>=1.49 in /usr/local/lib/python3.7/dist-packages (from geopy) (1.52)
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (5.5.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from plotly) (1.15.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packages (from plotly) (8.0.1)
```

From the above code we can say that, initially we need to install geopy library which is used to give the standard state names of India. It has its own Regex model where it checks if the following data that we pass is closest to one of the states of India. It will return that state. Also, we need to install plotly which is used to generate the India map
With all the data filled in.

```
import pandas as pd
from geopy.geocoders import Nominatim
def getRawLoc(data):
    try:
        location = geolocator.geocode(data, addressdetails=True)
    except:
        return
    if str(type(location)) == '<class \'geopy.location.Location\'>':
        if 'address' in location.raw and 'country' in location.raw['address'] and location.raw['address']['country'] == 'India':
            if 'state' in location.raw['address']:
                return location.raw['address']['state']
    return

state_df = pd.read_csv('/content/IPL_2021_tweets.csv', lineterminator='\n')
state_df = state_df['user_location'].value_counts().reset_index()
state_df['state_location'] = state_df['index'].apply(getRawLoc)
state_df = state_df.groupby('state_location').sum('user_location').reset_index().rename(columns={'user_location':'count'})
state_df
```

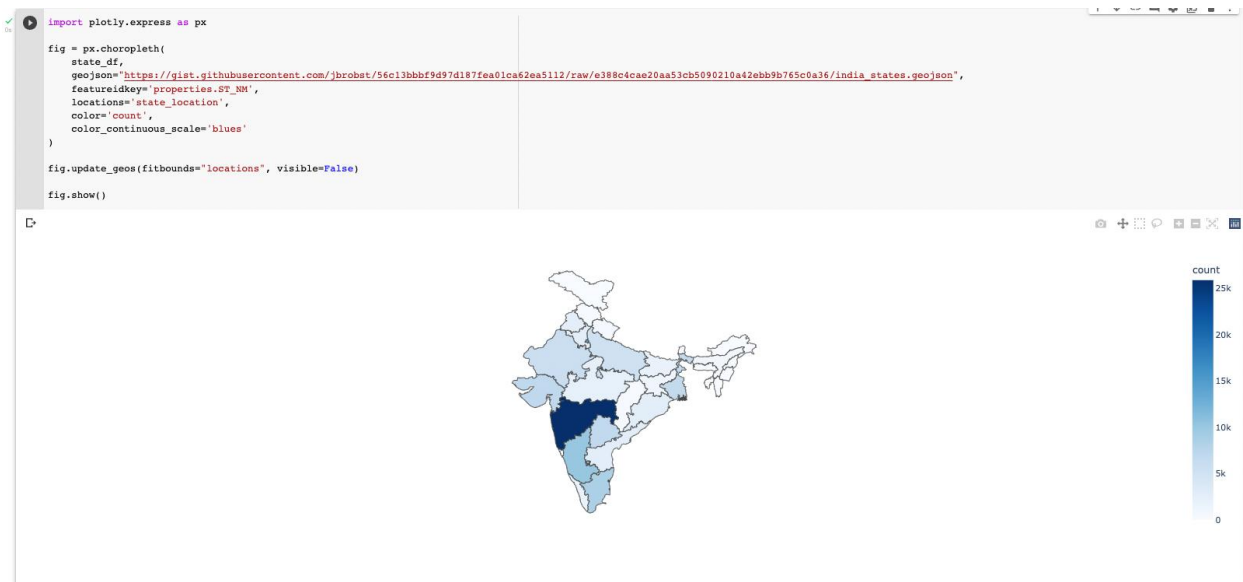
The above code is where we first generate the Pandas DataFrame from the IPL tweets csv file. Then we club all the similar data of user_location and create a new column with clubbed data count of each in rows time it appeared. Then we are iterating through each row of the element and getting its user location state. Here in the function, we are calling the api geolocator of geopy api which gives us the geopy location object, from which we extract only those location which are in India and have a dedicated state field in them.

Once we have the list of all the states in the variable `states_df` data frame we again group the data frame by the states and sum all their counts. Which finally gives us the total number of tweets that have appeared from different states in India.

The output generated is as follows:

| | state_location | count |
|----|--|-------|
| 0 | Andhra Pradesh | 2628 |
| 1 | Arunachal Pradesh | 2 |
| 2 | Assam | 801 |
| 3 | Bihar | 1776 |
| 4 | Chandigarh | 949 |
| 5 | Chhattisgarh | 305 |
| 6 | Dadra and Nagar Haveli and Daman and Diu | 10 |
| 7 | Delhi | 14281 |
| 8 | Goa | 268 |
| 9 | Gujarat | 6924 |
| 10 | Haryana | 2592 |
| 11 | Himachal Pradesh | 231 |
| 12 | Jammu and Kashmir | 652 |
| 13 | Jharkhand | 1101 |
| 14 | Karnataka | 10006 |
| 15 | Kerala | 1674 |

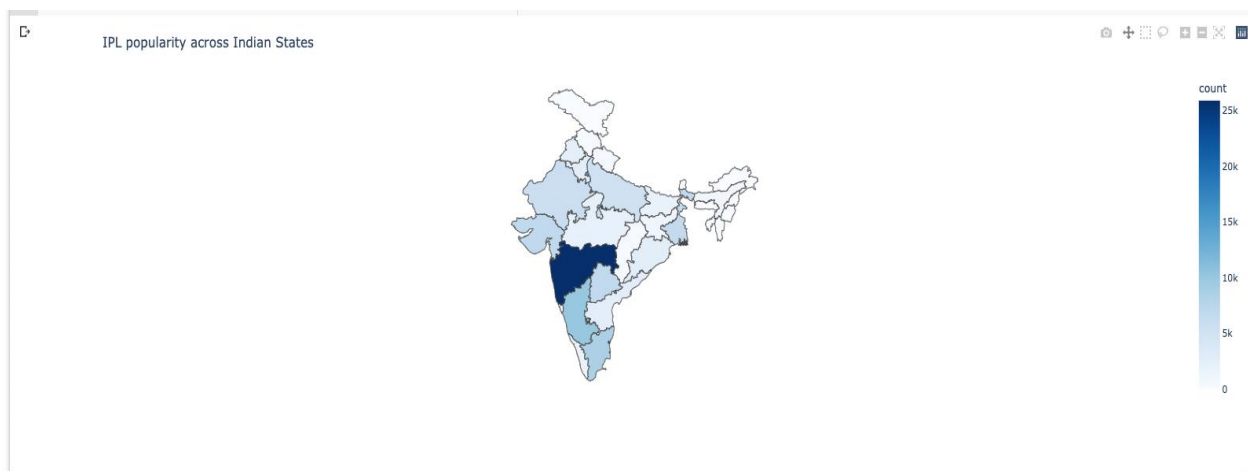
Below is the code we used to generate the map of India with different states. The geoJson param that we send as part of the api request to plotly links state point of the individual states of India to form a map. As we see from the below code that we are passing the param location as `state_location` which contains all the state name and color contains the count of each state.



2.3 : Results and Insights

From the geographic data snippet attached below, we can see that Maharashtra is the most active crowd in IPL, this also resemblance that the team has won 5 times IPL trophy which would have resulted in higher fan base. Also, one of the strange things we observed where the crowd in Karnataka are more interested in game than people in Tamil Nadu. But the team that is representing Karnataka has not yet won any IPL trophy compared to the team of Tamil Nadu which has won 4 times.

Also, we can see that the interest of the game in the eastern region(7 sisters states of India) of India is not that popular. This might be because there is no IPL team that represents them. So looking forward IPL can move some of the matches there or introduce a team from that region which can result significant increase in popularity towards the game.



Feature 3 : Determine most marketable team in for that year

3.1 : Overview

In here we are calculating the frequency with which each team is mentioned from the tweets that were sent in whole year and describing how this data can be used in future seasons.

3.2 : Implementation

We used pandas to export the csv data to pandas DataFrame format. Thereby we can access all the mined data easily.

```
ipltweetdata = pd.read_csv('/content/IPL_2021_tweets.csv', lineterminator='\n')
ipltweetdata.info()
tweets = ipltweetdata.text.apply(str)

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWarning: Columns (5,6,7,12) have
exec(code_obj, self.user_global_ns, self.user_ns)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252395 entries, 0 to 252394
Data columns (total 13 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   user_name            252388 non-null object
1   user_location        178878 non-null object
2   user_description     221018 non-null object
3   user_created         252393 non-null object
4   user_followers       252393 non-null float64
5   user_friends         252393 non-null object
6   user_favourites      252393 non-null object
7   user_verified        252393 non-null object
8   date                 252393 non-null object
9   text                 252393 non-null object
10  hashtags             231251 non-null object
11  source               252323 non-null object
12  is_retweet           252389 non-null object
dtypes: float64(1), object(12)
memory usage: 25.0+ MB
```

In here as you can see, we are reading the data from the CSV file in the format of the DataFrame format, where we can see we have different columns. Where we will be using only the text column for further analysis of data.

Using the text column from the dataframe, we clean the data to remove urls, any '#'s and any '@'s from the text data. After that, we convert all of the text data to lowercase for uniformity. The below text is after the cleaning steps mentioned above.

```
tweets_lowercase
confirmed rcb signs england iercarm pacer george garton as a replacement player for the ipl second phase in
'welcome to the rcb family george nowaroyalchallenger ipl2021 rcb wearechallengers playbold',
'news alert george garton joins rcb as their eighth overseas player for ipl2021 uae leg',
'royal challengers bangalore announce george garton as the replacement for kane richardson for the phase 2 o
'ipl2021 breaking news george garton joins rcb official announcement made by rcb',
'george garton is signed by rcb for ipl 2021ipl2021',
'its happening george garton to rcb ipl2021',
'rcb have signed george garton for ipl2021 phase 2',
'welcome to the team george playbold wearechallengers nowachallenger ipl2021',
'announcement talented allrounder from england george garton will join the rcb family for the rest of ipl202
'virat kohli is a lovely welcoming guy who is passionate about winning says rcb pacer kyle jamiesonviratkohl
'csk vs mi 2021 highlightsbumrahs 5 ferel clasico of iplcricket 19 ga via youtube ipl2021 cskvsmi csk mi ps4
'why virat struggle in england 2021 why anderson takes wicket of virat babarazam fawadalamcricket rcb englan
'new look of mahiifor remaining ipl matcheswhat you think about this lookipl ipl2021 cricket dhoni mahendras
'kohli gone dont worry he will score a century on oct 15 2021 and will bring it homercbtweets engvind ipl202
'india loosing all their wickets so their main batters j bumrah and m shami have enough time to settle thems
'ipl 2021 rcbs kyle jamieson terms virat kohli as lovely welcoming guy passionate about winning engvind indv
'jamieson on kohli he is a lovely guy i have played against him a couple of times and obviously he is quite
```

Now, we use CountVectorizer to get the count of words in the complete corpus. We also define a list of words for which we need to find the count which in this case are the teams.

```
[ ] vectorizer = CountVectorizer(encoding='latin-1', ngram_range=(1,3), min_df=20, stop_words='english')
vecs = vectorizer.fit_transform(tweets_lowercase)
feat_dict=vectorizer.vocabulary_

[ ] teams = ["chennai super kings", "csk",
"royal challengers bangalore", "rcb",
"delhi capital", "dc",
"mumbai indians", "mi",
"kolkata knight riders", "kkr",
"rajasthan royals", "rr",
"kings xi punjab", "kxip",
"sunrisers hyderabad", "srh"]
```

Using the CountVectorizer's feature dictionary, we make a list of tuples with key value pairs, as the team name and frequency.

```
[14] team_vocab = []
for team in teams:
    team_vocab.append((team, feat_dict.get(team)))

[15] team_vocab

[('chennai super kings', 5302),
('csk', 6382),
('royal challengers bangalore', 20004),
('rcb', 19003),
('delhi capital', 7289),
('dc', 6978),
('mumbai indians', 16085),
('mi', 15290),
('kolkata knight riders', 13689),
('kkr', 13267),
('rajasthan royals', 18743),
('rr', 20078),
('kings xi punjab', 13240),
('kxip', 13828),
('sunrisers hyderabad', 22573),
('srh', 22049)]
```

We convert this list to a dictionary which has team names only with other entries added into them, we further convert the dictionary to a pandas Dataframe object to make it easier to plot.

```
[16] team_dict = dict()
for i in range(0, len(team_vocab), 2):
    team_dict[team_vocab[i][0]] = int(team_vocab[i][1] or 0) + int(team_vocab[i+1][1] or 0)

[17] team_dict

{'chennai super kings': 11684,
'delhi capital': 14267,
'kings xi punjab': 27068,
'kolkata knight riders': 26956,
'mumbai indians': 31375,
'rajasthan royals': 38821,
'royal challengers bangalore': 39007,
'sunrisers hyderabad': 44622}
```

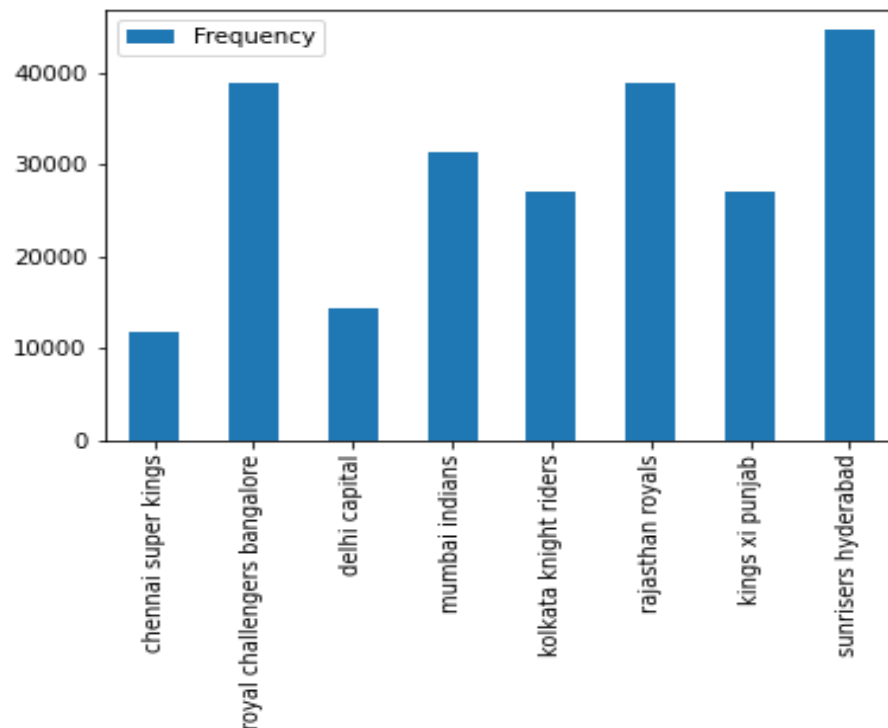
```
✓ [23] teamtable = pd.DataFrame.from_dict(team_dict, orient='index')
0s      teamtable.columns = ['Frequency']

✓ [24] print(teamtable)
0s
```

| | Frequency |
|-----------------------------|-----------|
| chennai super kings | 11684 |
| royal challengers bangalore | 39007 |
| delhi capital | 14267 |
| mumbai indians | 31375 |
| kolkata knight riders | 26956 |
| rajasthan royals | 38821 |
| kings xi punjab | 27068 |
| sunrisers hyderabad | 44622 |

3.3 : Results and Insights

We get the final resulting bar graph as below:



From the above graph, we can see that the team Sunrisers Hyderabad has the maximum number of mentions in the tweets. The IPL organizers can use this data in the next season to determine marketing strategies and budgets. The team owners themselves can use this data to make changes to their marketing approaches.

Feature 4 : Determine The Most Marketable Player For Each Team

4.1 : Overview

In this insight, we are trying to understand how we can identify highly marketable players. There is a strong fan following for local teams and players irrespective of the country have a strong following. Identifying right marketable players is very significant for sponsors and advertisers as these players tend to have a strong and religious fan bases across India and even across the world. Hence, we analyze the tweets to determine which are some of the marketable players for each team and determine the most marketable players in the IPL for 2021.

4.2 : Implementation and Results

To determine this insight, the tweets were first segregated by each of the 8 IPL teams.

```
teams = ["chennai super kings", "csk",
"royal challengers bangalore", "rcb",
"delhi capital", "dc",
"mumbai indians", "mi",
"kolkata knight riders", "kkr",
"rajasthan royals", "rr",
"kings xi punjab", "kxip",
"sunrisers hyderabad", "srh"]
```

```
def returnTweet(text, item1, item2):
    if (item1 in text) or (item2 in text):
        return text
    else:
        return ""
```

```
ipl["csk"] = ipl.refine_text.apply(lambda x: returnTweet(str(x),"chennai super kings", "csk" ))
ipl["rcb"] = ipl.refine_text.apply(lambda x: returnTweet(str(x),"royal challengers bangalore", "rcb" ))
ipl["dc"] = ipl.refine_text.apply(lambda x: returnTweet(str(x),"delhi capital", "dc" ))
ipl["mi"] = ipl.refine_text.apply(lambda x: returnTweet(str(x),"mumbai indians", "mi" ))
ipl["kkr"] = ipl.refine_text.apply(lambda x: returnTweet(str(x),"kolkata knight riders", "kkr" ))
ipl["rr"] = ipl.refine_text.apply(lambda x: returnTweet(str(x),"rajasthan royals", "rr" ))
ipl["kxip"] = ipl.refine_text.apply(lambda x: returnTweet(str(x),"kings xi punjab", "kxip" ))
ipl["srh"] = ipl.refine_text.apply(lambda x: returnTweet(str(x),"sunrisers hyderabad", "srh" ))
```

For each set of tweets that are divided by teams, a corpus was created by combining all the tweets for that team.

```
csk_tweets = ipl["csk"].sum()
rcb_tweets = ipl["rcb"].sum()
dc_tweets = ipl["dc"].sum()
mi_tweets = ipl["mi"].sum()
kkr_tweets = ipl["kkr"].sum()
rr_tweets = ipl["rr"].sum()
kxip_tweets = ipl["kxip"].sum()
srh_tweets = ipl["srh"].sum()
```

A custom word cloud was generated for each of the 8 IPL Teams and some insights that were drawn regarding the players are as follows:

Feature 5 : Determine the IPL Tweet captivating trends depending on the number of tweets per day.

5.1 : Overview

In here we are calculating the tweets that were sent in whole year and describing how the human interfaces with social world starting from IPL players auction to the games. One of the unique features of the IPL 2021 was that the IPL happened among the midst of Covid pandemic which resulted in playing of half games in India and then shifting the other half to United Arab Emirates, which happened at the second half of the year. Usually, IPL happens in the early spring to summer whereas in the year 2021 the IPL started in early spring. But as the days went the number of Covid cases started rising and accidentally few players caught Covid which resulted in halting the IPL in India. Also, this was the time when the human sentiment was worse as India as a country was battered by the worst outbreak of covid and there was a larger human loss. Later, when the things got under control the BCCI (Board of Control of Cricketing in India) had taken massive steps to curb this outbreak. So, they shifted the IPL to UAE which happened in the second half of the year starting from September to October 2021.

5.2 : Implementation

Using our Twitter mining algorithm, we mined the all the tweets that were related to IPL and had put the data in the csv format. We used pandas to export the csv data to pandas DataFrame format. Thereby we can access all the mined data easily.

```

38 ipl=pd.read_csv('/content/IPL_2021_tweets.csv', lineterminator='\n')
ipl.info()

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWarning:
Columns (5,6,7,12) have mixed types.Specify dtype option on import or set low_memory=False.

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252395 entries, 0 to 252394
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   user_name              252388 non-null  object
1   user_location          178878 non-null  object
2   user_description       221018 non-null  object
3   user_created           252393 non-null  object
4   user_followers         252393 non-null  float64
5   user_friends           252393 non-null  object
6   user_favourites        252393 non-null  object
7   user_verified          252393 non-null  object
8   date                  252393 non-null  object
9   text                   252393 non-null  object
10  hashtags               231251 non-null  object
11  source                 252323 non-null  object
12  is_retweet             252389 non-null  object
dtypes: float64(1), object(12)
memory usage: 25.0+ MB

```

In here as you can see, we are reading the data from the CSV file in the format of the DataFrame format, where we can see we have different columns. Where we will be using only the date column for further analysis of data.


```

ipl['date'] = pd.to_datetime(ipl['date'],infer_datetime_format=True,errors='coerce')
ipl['tweet_date']=ipl['date'].dt.date
tweet_date=ipl['tweet_date'].value_counts().to_frame().reset_index().rename(columns={'index':'date','tweet_date':'count'})
tweet_date['date']=pd.to_datetime(tweet_date['date'],infer_datetime_format=True,errors='coerce')
tweet_date=tweet_date.sort_values('date',ascending=True)
tweet_date

```

| | date | count |
|-----|------------|-------|
| 6 | 2021-02-18 | 15813 |
| 15 | 2021-02-19 | 3147 |
| 25 | 2021-02-20 | 729 |
| 28 | 2021-02-21 | 659 |
| 34 | 2021-02-22 | 549 |
| ... | ... | ... |
| 26 | 2021-10-03 | 694 |
| 11 | 2021-10-04 | 5849 |
| 37 | 2021-11-03 | 423 |
| 1 | 2021-11-04 | 22191 |
| 42 | 2021-12-03 | 361 |

From the above code we can see that we are converting the IPL DataFrame date to datetime format and removing the data which are not in proper format. Then creating a new series of IPL['tweet_data'] which stores this data in the IPL DataFrame format.

Then we create a Tweet_date DataFrame object. Which it is created from IPL['tweet_date'] as base data then we filter the series containing counts of unique values and renaming the Columns of the new DataFrame with index being date and tweet_date being count. Finally, we again convert the Tweet_date['date'] series to datetime format and sort the result according to the starting date of the data.

As we can see that the final result is sorted, and the final data looks cleaner and more organized. Now we will feed the data to a graph to make look more valuable.

```

fig=go.Figure(go.Scatter(x=tweet_date['date'],
                        y=tweet_date['count'],
                        mode='markers+lines',
                        name="Submissions",
                        marker_color='dodgerblue'))

fig.update_layout(
    title_text='Tweets per Day : ({} - {})'.format(tweet_date['date'].sort_values().iloc[0].strftime("%d/%m/%Y"),
                                                    tweet_date['date'].sort_values().iloc[-1].strftime("%d/%m/%Y")),
    title_x=0.5)

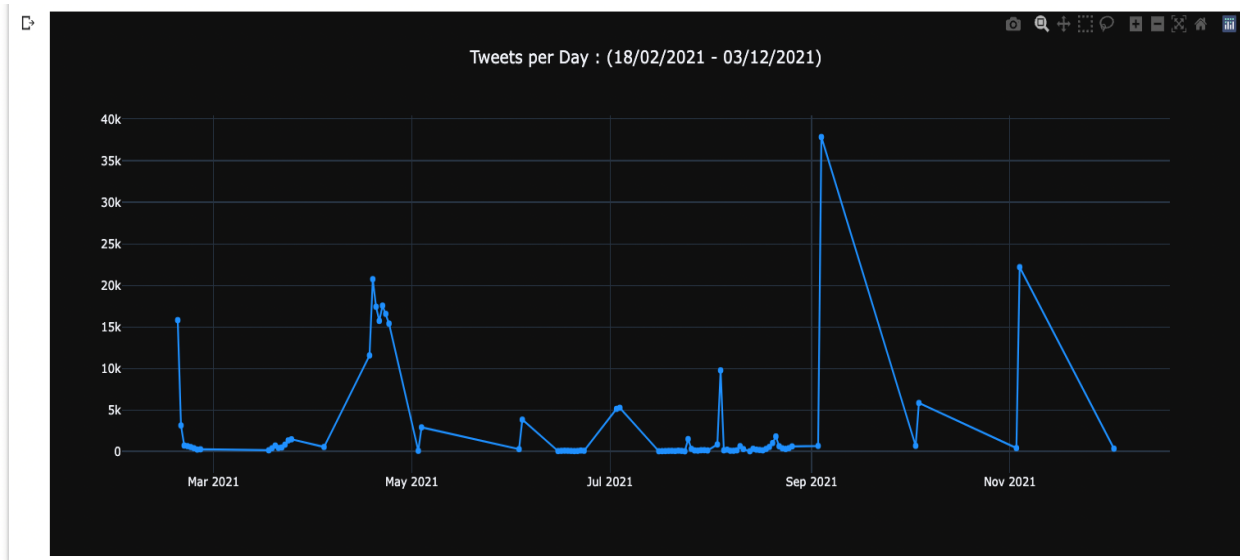
fig.show()

```

From the above code we can see that we are just creating a scatter plot with starting date as the start date of the Tweet_date['date'] series and end date of Tweet_date['date'] series.

5.3 : Results and Insights

From the graph below, we can see that the IPL trend tweets started picking up in early Feb as the IPL players auction happened where each player was auctioned to separate teams from where they will be playing for that teams in that year. As the days went by the IPL fever was nominal as the tweets were in the range of few hundreds then on the April 19, 2021, the tournament started and we can see that there is a very high spike in the data. Then as the days went the covid cases started increasing in India which as a result dropped the significant number of tweets related to IPL as well and on May4 the BCCI stopped the Tournament which gave a small spike as many were flocked to twitter on the announcement. Later, we can see that on August4 the IPL trading tweet was back as the BCCI announced it will continue the Tournament in UAE from Sept 4. As we can see the IPL Fever was back to public and the Tournament progressed till Nov4 which was the Final match.



Feature 6 : Determine the user sentiment based on tweets

6.1 : Overview

For this feature, we are going to perform sentiment analysis on tweets done by different users during the IPL match season in year 2021. Sentiment analysis is a technique used to determine whether the data is positive, negative, or neutral. The process of analyzing sentiments from tweets comes under “Classification”. This process involves identifying “patterns” from large set of data. In this project, machine learning techniques are used to extract the features and patterns from the data set of tweets. In our scenario, this is being done to capture the audience sentiments towards the game while they are watching and posting tweets on twitter. We have limited this sentiment analysis by focusing on polarity of the tweets.

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool and is adjusted according to the sentiments expressed on social media. It is also an open sourced tool to perform analysis. Since our analysis is also on twitter data, Vader is the perfect fit for analysis.

6.2 : Implementation

First, we will be installing the vaderSentiment package from the Pypi in our current environment using command pip install.

```
[ ] !pip install vaderSentiment

Requirement already satisfied: vaderSentiment in /usr/local/lib/python3.7/dist-packages (3.3.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from vaderSentiment) (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (2.10)
Requirement already satisfied: certifi<=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (2021.10.8)
Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (1.24.3)
```

Further the mined data was export to pandas' data frame format from the csv file available. This will help in using the mined data properly for analysis. The next was to analyze the data and create the copy of the tweet: text column so that we can make changes and perform exploratory data analysis which will help in visualizing the analysis properly.

```
#Creating a copy of text so that we can make changes and perform sentiment analysis
df['senttext'] = df['text']
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252395 entries, 0 to 252394
Data columns (total 14 columns):
#   Column              Non-Null Count  Dtype
---  -
0   user_name            252388 non-null  object
1   user_location        178878 non-null  object
2   user_description     221018 non-null  object
3   user_created         252393 non-null  object
4   user_followers       252393 non-null  float64
5   user_friends         252393 non-null  object
6   user_favourites      252393 non-null  object
7   user_verified        252393 non-null  object
8   date                 252393 non-null  object
9   text                 252393 non-null  object
10  hashtags             231251 non-null  object
11  source               252323 non-null  object
12  is_retweet           252389 non-null  object
13  senttext             252393 non-null  object
dtypes: float64(1), object(13)
memory usage: 27.0+ MB
```

In next step we converted all the data present in our data frame to lowercase. This will help in removing stop words from the text in the next step for analysis.

```
#cleaning data
df = df.apply(lambda x: x.astype(str).str.lower())
stop_words = stopwords.words('english')
df.text = df.text.apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))
df.text = df.text.apply(lambda x: ' '.join(re.sub("[@A-Za-z0-9+]|#[A-Za-z0-9+]", " ", x).split()))
df.text = df.text.apply(lambda x: ' '.join(re.sub("[\.\,!\?;\:;\-\\=", " ", x).split()))
df.text = df.text.apply(lambda x: ' '.join(re.sub(r'http\S+', '', x).split()))
```

Since our main analysis was based on tweets done by user, we will be performing data cleaning in that particular column only. In the above step, we are trying to remove all the stop words, any symbols, or links present in the tweets. Some tweets also contain emojis in them as shown below.

```
try:
    # UCS-4
    e = re.compile(u'[\U00010000-\U0010ffff]')
except re.error:
    # UCS-2
    e = re.compile(u'[\ud800-\udbff][\udc00-\udfff]')
emojis = []
for x in df.text:
    match = e.search(x)
    if match:
        emojis.append(match.group())

dfe = pd.DataFrame(emojis, columns=['text'])
pd.Series(' '.join(dfe['text']).lower().split()).value_counts()[0:10]
```

```
5885
5335
3004
2648
1850
1510
1468
1435
1381
1304
dtype: int64
```

So, to make data cleaner we have removed the emojis present in all tweets.

```
df = df.astype(str).apply(lambda x: x.str.encode('ascii', 'ignore').str.decode('ascii'))
```

Since we are done with all data cleaning, the next step was to apply the VADER sentiment analyzer on our tweets to measure the user sentiment towards the match.

```
[33] analyser = SentimentIntensityAnalyzer()

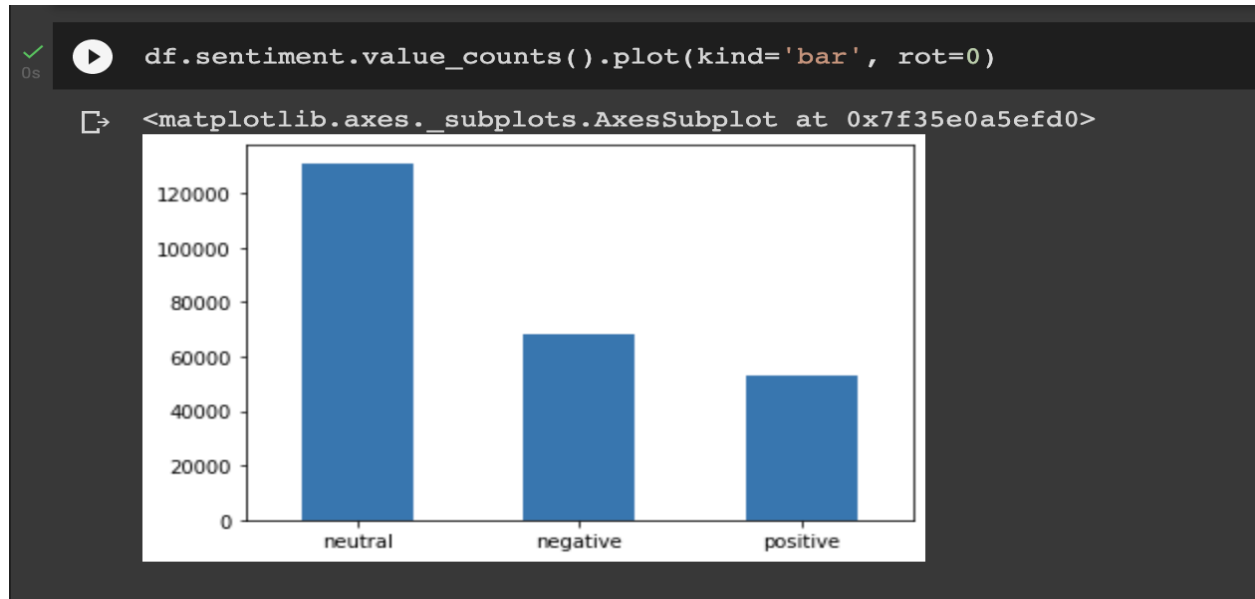
df['sentiment_score'] = df['senttext'].apply(lambda x: analyser.polarity_scores(str(x)))

def sentiment_func(sentiment):
    for k,v in sentiment.items():
        if (sentiment['compound'] >= 0.5 ) == True:
            return 'positive'
        elif (sentiment['compound'] == 0.0) == True:
            return 'neutral'
        else:
            return 'negative'

df['sentiment'] = df['sentiment_score'].apply(sentiment_func)
```

In the above code we are calculating polarity of each tweet done by the user and storing them in a new column. The new columns will have the polarity in the form of list stating the compound, negative, positive, and neutral score for that particular tweet. Then we are iterating over that list and defining whether the tweets sentiment based on the compound value.

6.3 : Results and Insights



The bar graph plotted above shows us the sentiments of user tweets. Following are the insights:

1. We can see that nearly 50% of the tweets are neutral, the reason behind this could be that many tweets contain just the score updates and hence neutral.
2. Negative tweets are more than positive tweets by around 20,000 tweets. There could be two main reasons behind this:
 - i) Some of the high expected players were not performing up to their potential.
 - ii) The COVID-19 cases were increasing drastically in India and the government was not stopping the IPL and gave more preference to it, which made users agitated and tweet negative about IPL in general and not teams.

To check if the above insight made regarding the negative tweets, we decided to check user sentiment as per the teams, which would give us a clear perspective regarding the reason behind the negative tweets.

Feature 7 : Determine user sentiment as per teams.

7.1 : Overview

For this feature, we are going to perform sentiment analysis on tweets done by different users during the IPL match season in year 2021. Sentiment analysis is a technique used to determine whether the data is positive, negative, or neutral. The process of analyzing sentiments from tweets comes under “Classification”. This process involves identifying “patterns” from large set of data. In this project, machine learning techniques are used to extract the features and patterns from the data set of tweets. In our scenario, this is being done to capture the audience sentiments towards the game while they are watching and posting tweets on twitter. We have limited this sentiment analysis by focusing on polarity of the tweets.

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool and is adjusted according to the sentiments expressed on social media. It is also an open sourced tool to perform analysis. Since our analysis is also on twitter data, Vader is the perfect fit for analysis.

7.2 : Implementation

First, we will be installing the vaderSentiment package from the Pypi in our current environment using command pip install.

```
[ ] !pip install vaderSentiment

Requirement already satisfied: vaderSentiment in /usr/local/lib/python3.7/dist-packages (3.3.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from vaderSentiment) (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (2021.10.8)
Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (1.24.3)
```

Further the mined data was export to pandas’ data frame format from the csv file available. This will help in using the mined data properly for analysis. The next was to analyze the data and create the copy of the tweet: text column so that we can make changes and perform exploratory data analysis which will help in visualizing the analysis properly.

```
#Creating a copy of text so that we can make changes and perform sentiment analysis
df['senttext'] = df['text']
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252395 entries, 0 to 252394
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ---
 0   user_name           252388 non-null object
 1   user_location       178878 non-null object
 2   user_description    221018 non-null object
 3   user_created        252393 non-null object
 4   user_followers      252393 non-null float64
 5   user_friends        252393 non-null object
 6   user_favourites     252393 non-null object
 7   user_verified       252393 non-null object
 8   date                252393 non-null object
 9   text                252393 non-null object
10   hashtags            231251 non-null object
11   source              252323 non-null object
12   is_retweet          252389 non-null object
13   senttext            252393 non-null object
dtypes: float64(1), object(13)
memory usage: 27.0+ MB
```

In next step we converted all the data present in our data frame to lowercase. This will help in removing stop words from the text in the next step for analysis.

```
#cleaning data
df = df.apply(lambda x: x.astype(str).str.lower())
stop_words = stopwords.words('english')
df.text = df.text.apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))
df.text = df.text.apply(lambda x: ' '.join(re.sub("([A-Za-z0-9]+)|#[A-Za-z0-9]+", " ", x).split()))
df.text = df.text.apply(lambda x: ' '.join(re.sub("[\.\,\!\?;\:\-\=\]", " ", x).split()))
df.text = df.text.apply(lambda x: ' '.join(re.sub(r'http\S+', '', x).split()))
```

Since our main analysis was based on tweets done by user, we will be performing data cleaning in that particular column. In the above step, we are trying to remove all the stop words, any symbols, or links present in the tweets. Some tweets also contain emojis in them as shown below. So, to make data cleaner we have removed the emojis present in all tweets.

```
df = df.astype(str).apply(lambda x: x.str.encode('ascii', 'ignore').str.decode('ascii'))
```

Since we are done with all data cleaning, the next step was to apply the VADER sentiment analyzer on our tweets to measure the user sentiment towards the match. We first need to categorize tweets according to the teams playing in match. We created a list of team names and searched for those teams in tweets and created a new column for categorizing them as per teams. Since, tweets can also contain the short names of teams, we considered team as well for categorization.

```
[51] teams = ["chennai super kings", "csk",
             "royal challengers bangalore", "rcb",
             "delhi capital", "dc",
             "mumbai indians", "mi",
             "kolkata knight riders", "kkr",
             "rajasthan royals", "rr",
             "kings xi punjab", "kxip",
             "sunrisers hyderabad", "srh"]

[99] def category(a):
      for i in teams:
          if i in a:
              return i
      return "others"

ipltweetdata["teams"] = ipltweetdata.text.apply(lambda x: category(str(x)))
```

To make column data consistent, we had to modify all the nicknames to the original names after the above step. This helped in maintaining uniqueness for the team names.

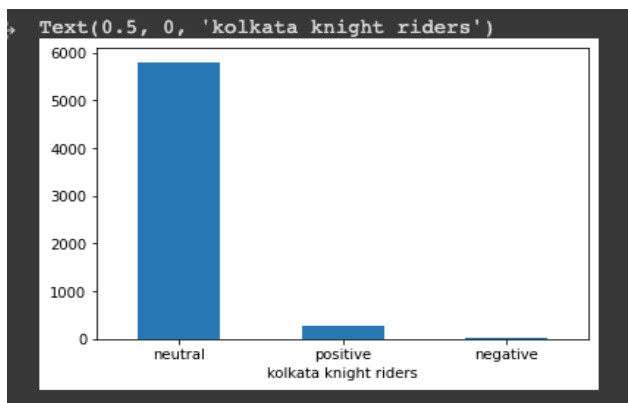
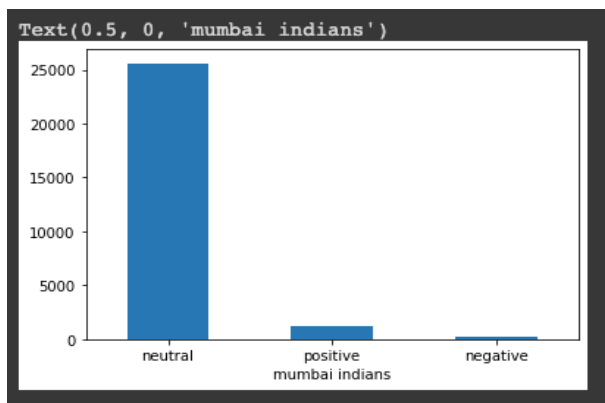
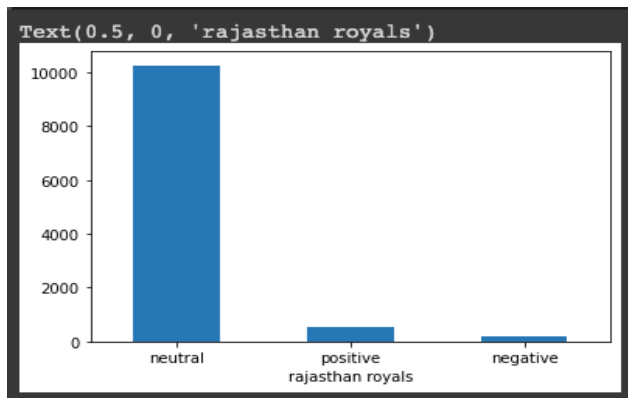
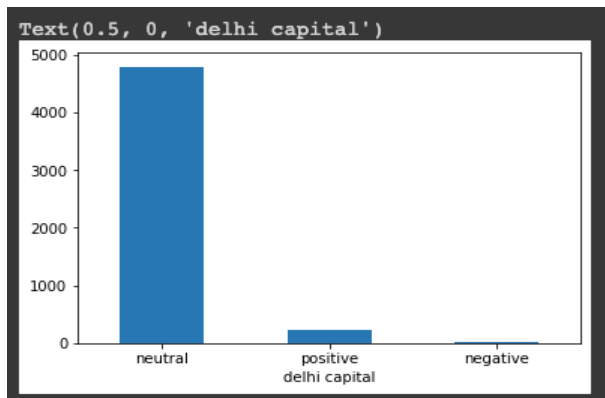
```
[66] ipltweetdata.loc[ipltweetdata["teams"] == "rr", "teams"] = "rajasthan royals"
      ipltweetdata.loc[ipltweetdata["teams"] == "dc", "teams"] = "delhi capital"
      ipltweetdata.loc[ipltweetdata["teams"] == "kkr", "teams"] = "kolkata knight riders"
      ipltweetdata.loc[ipltweetdata["teams"] == "mi", "teams"] = "mumbai indians"
      ipltweetdata.loc[ipltweetdata["teams"] == "csk", "teams"] = "chennai super kings"
      ipltweetdata.loc[ipltweetdata["teams"] == "rcb", "teams"] = "royal challengers bangalore"
      ipltweetdata.loc[ipltweetdata["teams"] == "kxip", "teams"] = "kings xi punjab"
      ipltweetdata.loc[ipltweetdata["teams"] == "srh", "teams"] = "sunrisers hyderabad"
```

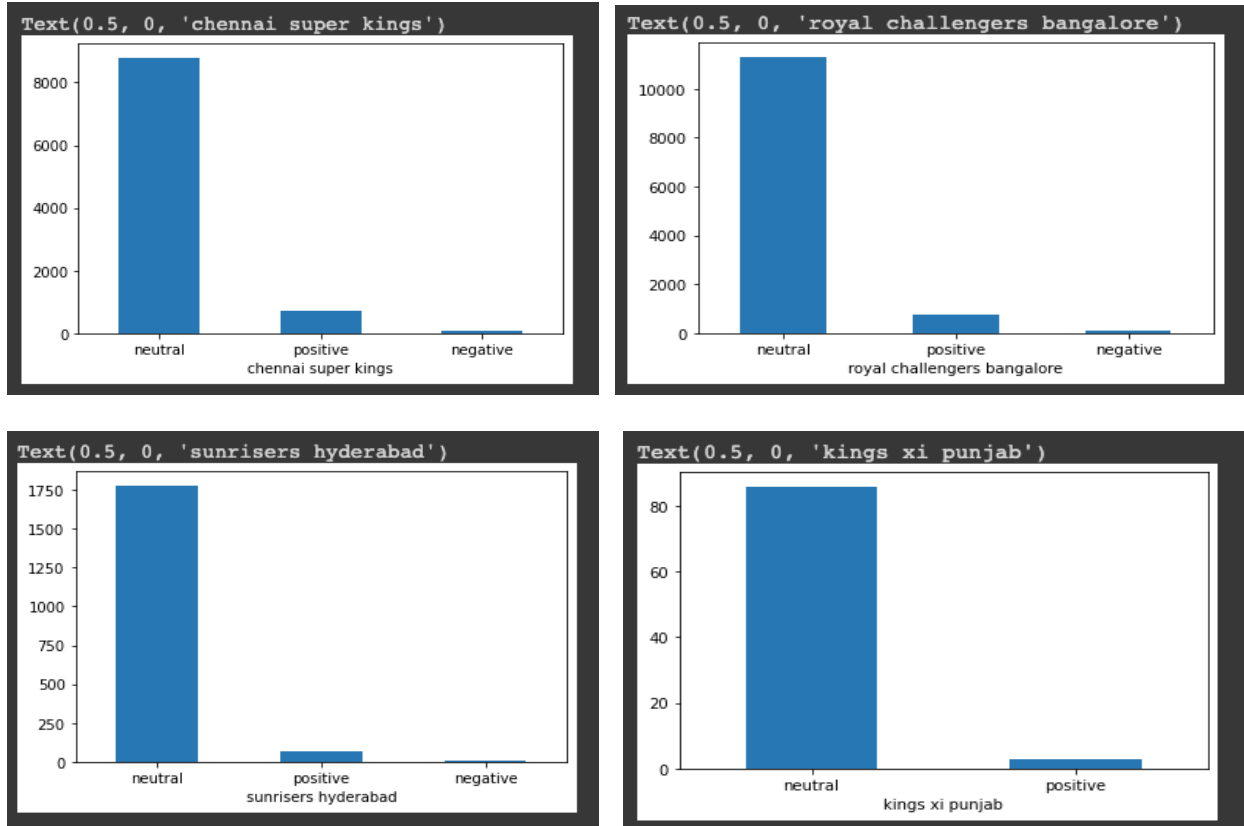
The next step was to apply sentiment analyzer for calculating polarity of each tweet done by the user as per teams and storing them in a new column. The new columns will have the polarity in the form of list stating the compound, negative, positive, and neutral score for that tweet. Then we are iterating over that list and defining whether the tweets sentiment based on the compound value.

```
[89] analyser = SentimentIntensityAnalyzer()
def sentiment_func(sentiment):
    # print(s['pos'])
    for k,v in sentiment.items():
        if (k == 'pos' or k == 'neg' or k == 'neu') == True:
            if (sentiment['pos'] > 0.5 and sentiment['neg'] < 0.5 and sentiment['neu'] < 0.5) == True:
                return 'positive'
            elif (sentiment['pos'] < 0.5 and sentiment['neg'] > 0.5 and sentiment['neu'] < 0.5) == True:
                return 'negative'
            elif (sentiment['pos'] < 0.5 and sentiment['neg'] < 0.5 and sentiment['neu'] > 0.5) == True:
                return 'neutral'
```

7.3 : Results and Insights

After this we plotted bar graph for each team, evaluating the sentiment analysis of user tweets for individual teams.





The bar graphs plotted above shows us the sentiments of user tweets as per every individual team. Following are the insights:

1. We can see that all the teams have neutral tweets the most, the reason behind this could be as stated in the previous feature that many tweets contain just the score updates and hence it is neutral.
2. It is followed by positive tweets and then negative tweets. We barely see any negative tweets for every team. Thus, the point made in the previous feature is correct that, “the COVID-19 cases were increasing drastically in India and the government was not stopping the IPL and gave more preference to it, which made users agitated and tweet negative about IPL in general and not teams.”

So, to conclude from Feature 6 and Feature 7: The IPL in general got a lot of backlash, hatred and negativity (from feature 6 graph), but the individual teams were not the once to receive a lot of negativity.

Section 5 : References

Python Documentation: <https://docs.python.org/3/>

2. Pandas API Reference : <https://pandas.pydata.org/pandas-docs/stable/reference/index.html>
 3. GeoPandas : <https://geopandas.org/en/stable/>
 4. Vader Sentiment :
 - (i) <https://github.com/cjhutto/vaderSentiment>
 - (ii) Simplifying Sentiment Analysis using VADER in Python (on Social Media Text) : <https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f>
 5. The dataset from Kaggle: https://www.kaggle.com/datasets/kaushiksuresh147/ipl2020-tweets?select=IPL_2021_tweets.csv
 6. Country Converter :
 - (i) <https://pycountry-convert.readthedocs.io/en/latest/>
 - (ii) https://github.com/konstantinstadler/country_converter
 7. GeoPy: <https://geopy.readthedocs.io/en/stable/>
 8. plotPy : <https://plotly.com/python/>
 9. NLTK Package Documentation: <https://www.nltk.org/api/nltk.html>
-